Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures.

AUTHOR(S):

Zhao, Yang; Hayashida, Morihiro; Akutsu, Tatsuya

CITATION:

Zhao, Yang ...[et al]. Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures.. BMC bioinformatics 2010, 11 (Suppl 11): S4.

ISSUE DATE:

2010-12-14

URL:

http://hdl.handle.net/2433/134588

Zhao et al. BMC Bioinformatics 2010, 11(Suppl 11):S4
http://www.biomedcentral.com/1471-2105-11-S11

BMC Bioinformatics

RESEARCH Open Access

Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures

Yang Zhao, Morihiro Hayashida, Tatsuya Akutsu*

From The 21st International Conference on Genome Informatics (GIW2010)
Hangzhou, People's Republic of China. 16-18 December 2010

Abstract

Background: A bisection-type algorithm for the grammar-based compression of tree-structured data has been proposed recently. In this framework, an elementary ordered-tree grammar (EOTG) and an elementary unordered-tree grammar (EUTG) were defined, and an approximation algorithm was proposed.

Results: In this paper, we propose an integer programming-based method that finds the minimum context-free grammar (CFG) for a given string under the condition that at most two symbols appear on the right-hand side of each production rule. Next, we extend this method to find the minimum EOTG and EUTG grammars for given ordered and unordered trees, respectively. Then, we conduct computational experiments for the ordered and unordered artificial trees. Finally, we apply our methods to pattern extraction of glycan tree structures.

Conclusions: We propose integer programming-based methods that find the minimum CFG, EOTG, and EUTG for given strings, ordered and unordered trees. Our proposed methods for trees are useful for extracting patterns of glycan tree structures.

Background

Data compression is useful because it can help reduce the consumption of expensive resources such as hard disks. To date, many methods such as Huffman coding, arithmetic coding, etc. have been proposed to solve problems of data compression. Data compression is also useful for the analysis of biological data. Li et al. proposed the universal similarity metric (USM), and approximated the dissimilarity using compression sizes. They applied a compression algorithm to unaligned mitochondrial genomes, and obtained a phylogeny that was consistent with the commonly accepted one [1]. Similarly, protein tertiary structures and metabolic networks were compressed, and their similarities were measured [2,3]. Grammar-based compression, which is a typical data-compression method, seeks a small grammar to generate a given string, as it is well known that it is NP-hard to find the smallest context-free grammar (CFG). However, in recent years, several polynomial time algorithms have been proposed to approximate the smallest grammar for the input data within a factor of $O(\log(n/m))$, where n and m are the sizes of the input data and the smallest grammar [4-6], respectively. These algorithms can be used to compress biological data such as DNA, RNA, and amino acid sequences. However, there exist a large amount of tree-structured biological data (e.g., glycan, etc.). Therefore, it is necessary to develop methods to compress tree-structured data. Recent approaches show that it is feasible to extend the grammar-based compression to the tree-structure data [7-9]. However, these methods neither output the

* Correspondence: takutsu@kuicr.kyoto-u.ac.jp
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto, 611-0011, Japan
Full list of author information is available at the end of the article

minimum grammar, nor they achieve a guaranteed approximation ratio.

In this paper, we propose an integer programming (IP)-based method that finds the minimum CFG for a given string under the condition that at most two symbols appear on the right-hand side of each production rule. Next, we extend this method to find the minimum elementary ordered-tree grammar (EOTG) and elementary unordered-tree grammar (EUTG) for given ordered and unordered trees. To the best of our knowledge, these are the first methods that can find the minimum size grammars for strings, ordered trees, and unordered trees.

It is possible to compress ordered trees by transforming them into Euler strings [10], and by applying existing grammar-based string-compression algorithms to the strings. Such an approach may achieve better compression performances. However, there do not always exist tree grammars corresponding to string grammars derived by the approach. Our objective is not only to compress trees but also to extract features and patterns from input trees. Therefore, we need to develop methods for finding minimum tree grammars. The organization of the paper is as follows. First, we give an IP-based method for sequence data compression using CFG. Second, we review an *elementary ordered tree grammar* (EOTG) [10] for ordered rooted tree compression, and extend this IP-based method to the tree compression problem. Then, we also review an *elementary unordered tree grammar* (EUTG) [10] for unordered rooted tree compression, and extend the above mentioned IP-based method for unordered trees. Furthermore, we conduct some computational experiments, apply the proposed methods to glycan tree-structure data, and extract tree patterns from generated production rules of simple EUTGs. Finally, we conclude with future work.

IP formulation for strings

Minimum CFG problem

We use a simple context-free grammar (CFG) for string and text compression. CFG is defined as 4-tuple (Σ, Γ, S, Δ), where Σ is a set of terminal symbols (denoted by a lower-case letter), Γ is a set of nonterminal symbols (denoted by an upper-case letter), S is a start symbol in Γ, and Δ is a set of production rules. The *size* of CFG is defined as the total number of letters appearing on the RHSs (Right-Hand Sides) of production rules. Two CFGs G_1 and G_2 are said to be *equivalent* if G_1 generates the same set of strings as G_2 does. We only consider CFGs consisting of the following types of production rules:

- $A \rightarrow a$,
- $A \rightarrow BC$.

We call this CFG *a simple CFG*. We can show that any CFG of size m can be transformed into an equivalent, simple CFG of size $3m$.

The smallest grammar problem is thus defined as the problem of finding the smallest grammar that generates a given string [4].

Minimum CFG

Input: String $s = s_1 s_2 \ldots s_n$ and integer m.

Output: Simple CFG with m nonterminal symbols that generates s only.

Transformation to IP

We use the number of nonterminal symbols m instead of the size of the grammar because the number of terminal symbols appearing in production rules of $A \rightarrow a$ is constant for the given string. In order to solve this minimum CFG problem, we propose an IP-based method as follows. We transform this problem into the following integer program, where $x_{1,n} = 1$ holds iff there exists a required CFG G.

Maximize $x_{1,n}$

Subject to $x_{i,i} = 1$ for all i=1,...,n

$$x_{i,j} \leq \sum_{k=i}^{j-1} y_{i,k,j} \quad \text{for all } 1 \leq i < j \leq n \tag{2}$$

$$y_{i,k,j} \leq \frac{1}{2}(x_{i,k} + x_{k+1,j}) \quad \text{for all } 1 \leq i \leq k < j \leq n \tag{3}$$

$$z_u \geq \frac{1}{n} \sum_{i,j:s_{i,j}=u} x_{i,j} \quad \text{for all distinct substrings } u \text{ of } s \tag{4}$$

$$\sum_u z_u = m \tag{5}$$

In the above equations, each variable of $x_{i,j}$, $y_{i,k,j}$, and z_u takes either 0 or 1. Each $x_{i,j}$ corresponds to substring $s_{i,j} = s_i s_{i+1} \ldots s_j$, and $x_{i,j} = 1$ iff there exists a nonterminal symbol $A_{i,j}$ in G that generates $s_{i,j}$. $y_{i,k,j} = 1$ iff both of $x_{i,k} = 1$ and $x_{k+1,j} = 1$ hold. It means that $s_{i,j}$ can be generated by concatenating $s_{i,k}$ and $s_{k+1,j}$ that are generated from nonterminal symbols $A_{i,k}$ and $A_{k+1,j}$, respectively. $z_u = 1$ iff there exists a nonterminal symbol in G that generates a substring u of s. The meaning of each (in)equality is as follows:

(**1**) each $s_{i,i}$ (= s_i) must be generated,

(**2,3**) if $A_{i,j}$ appears in G, $s_{i,j}$ must be generated, that is, for at least some k, both of $s_{i,k}$ and $s_{k+1,j}$ must be generated and the production rule $A_{i,j} \rightarrow A_{i,k}A_{k+1,j}$ must appear in G,

(**4**) $A_{i,j}$ and $A_{i',j'}$, are identified if both generate the same substring u, and

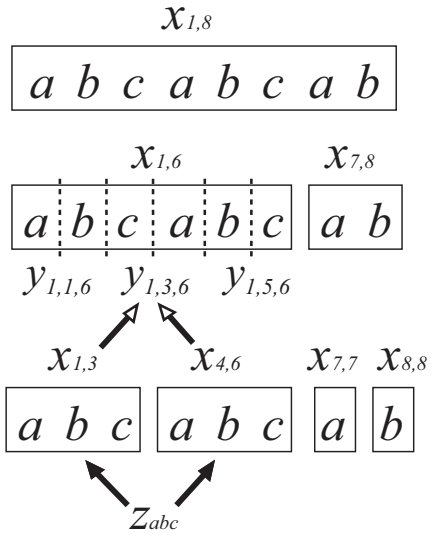

Figure 1 Illustration of the minimum CFG for a string "abcabcab" In the minimum CFG, the substring "abc" is generated from a nonterminal symbol A_{abc}. Then, $z_{abc} = 1$, $x_{1,3} = 1$, and $x_{4,6} = 1$. To generate "abcabc", that is, $x_{1,6} = 1$, at least one variable of $y_{1,k,6}$ ($k = 1,...,5$) must be 1, that is, the substring $s_{1,6}$ must be divided into $s_{1,k}$ and $s_{k+1,6}$. Here, $y_{1,3,6} = 1$ because $x_{1,3} = x_{4,6} = 1$. Then, the production rule $A_{abcabc} \rightarrow A_{abc}A_{abc}$ appears.

(**5**) the number of nonterminal symbols used in G must be m.

Figure 1 shows an example of the above IP formulation for the string "abcabcab". For this example, the following grammar is constructed from a solution of IP:

$A_{1,8} \rightarrow A_{1,6}A_{7,8}$,
$A_{1,6} \rightarrow A_{1,3}A_{4,6}$,
$A_{1,3} \rightarrow A_{1,2}A_{3,3}$,
$A_{4,6} \rightarrow A_{4,5}A_{6,6}$,
$A_{7,8} \rightarrow A_{7,7}A_{8,8}$,
....

On the other hand, we have $A_{abcabcab} = A_{1,8}$, $A_{abc} = A_{1,3} = A_{4,6}$, etc. Therefore, we finally have:

$A_{abcabcab} \rightarrow A_{abc}A_{abc}A_{ab}$,
$A_{abc} \rightarrow A_{ab}A_c$,
$A_{ab} \rightarrow A_aA_b$,
$A_a \rightarrow a$,
$A_b \rightarrow b$,
$A_c \rightarrow c$.

IP formulation for ordered trees

Minimum EOTG problem

We use a simple elementary ordered tree grammar (EOTG) [10] for rooted tree compression. In this grammar, a tree can contain a vertex called *a tag*. A tag indicates that another tree at the root can be attached to it. We assume that there is at most one tag in such a tree.

A simple EOTG (SEOTG) is defined as 4-tuple (Σ,Γ, S, Δ), where Σ is a set of terminal symbols, Γ is a set of nonterminal symbols; each edge of the trees has either a terminal or a nonterminal symbol; S is a start symbol in Γ, and Δ is a set of production rules (R1u,t), (R2u,t,t'), and (R3u,t), as in Figure 2. (R1u) ((R1t)) denotes a rule when an untagged (tagged) edge of nonterminal symbol A is replaced with an untagged (tagged) edge of terminal symbol a. (R2u,t,t') denotes a rule when an edge of a nonterminal symbol A is replaced with a tree that contains the upper endpoints of edges of nonterminal symbols B and C as the root, and the lower endpoints as two children. (R3u,t) denotes a rule when an edge of A is replaced with a tree in which the root is the upper endpoint of an edge of B, and the lower endpoint is the upper endpoint of an edge of C. We can show that any EOTG of size m can be transformed into an equivalent SEOTG of size $3m$.

Within the class of SEOTGs, we can transform the minimum grammar problem into the IP. For this purpose, we define the minimum SEOTG problem as follows.

Minimum SEOTG

Input: Rooted ordered tree $T(V, E)$ and integer m, where V is a set of vertices and E is a set of labeled edges.

Output: Simple EOTG with m nonterminal symbols that generates only T.

Transformation to IP

A subtree of $T(V, E)$ can be represented as a rooted ordered tree $T_{i,t,h,k}$ with a root i, a tag t, a left-most child h, and a right-most child k of i (Figure 3), where the tree

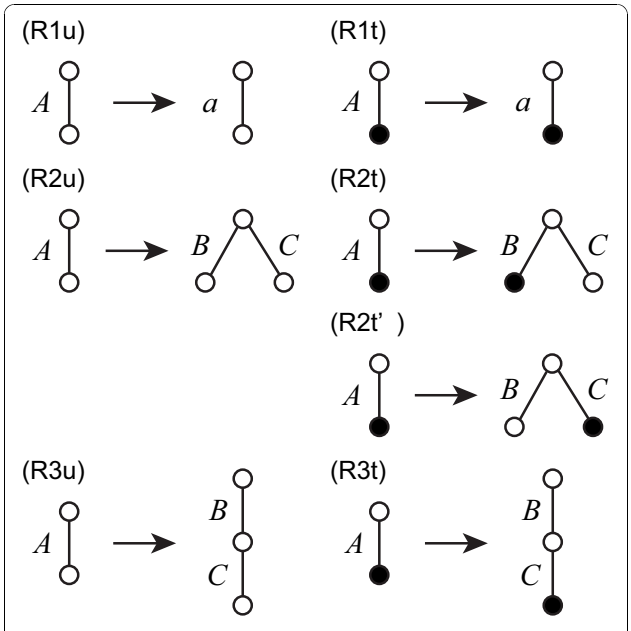

Figure 2 Production rules of simple EOTG A black circle denotes a tag.

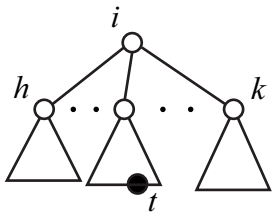

Figure 3 Example of ordered tree $T_{i,t,h,k}$

includes all the children of i between h and k in $T(V, E)$. If a subtree does not contain any tag, we introduce $\in$ that is not included in V, and represent it as $T_{i,\lfloor,h,k}$. It is obvious from the production rules of simple EOTGs that it is sufficient to consider only such subtrees $T_{i,t,h,k}$ for $T(V, E)$ because (R2u,t) denotes a rule to horizontally divide a tree into two trees at the root, and (R3u,t) denotes a rule to vertically divide a tree into two trees at an internal vertex that becomes a tag. Let $ch(i) = (lch(i),..., rch(i))$ denote a sequence of all the children of i in $T(V, E)$, $lch(i)$ is the left-most child, and $rch(i)$ is the right-most child. Without loss of generality, we assume that $i_1 \le ... \le i_k$ for $ch(i) = (i_1, ..., i_k)$. We suppose that the root of $T(V, E)$ is 1. Then, this problem can be transformed into the following integer program, where $x_{1\lfloor,lch(1),rch(1)} = 1$ holds iff there exists a required EOTG G.

Maximize $x_{1\lfloor,lch(1),rch(1)}$

Subject to

$$x_{i,\in,j,j} = 1 \quad \text{for all } i, j \in ch(i)\ (|\, ch(j) \models 0) \qquad (1u)$$

$$x_{i,j,j,j} = 1 \quad \text{for all } i, j \in ch(i)\ (|\, ch(j) \models 0) \qquad (1t)$$

$$x_{i,\in,h,k} \le \sum_{l=h}^{k-1} y^{ho}_{i,\in,h,l,k} + \sum_{t \in I(T_{i,\in,h,k})} y^{ve}_{i,\in,h,k,t} \quad \text{for all } i, h \le k \in ch(i) \qquad (2-3u)$$

$$y^{ho}_{i,\in,h,l,k} \le \frac{1}{2}(x_{i,\in,h,l} + x_{i,\in,l} + 1, k) \qquad (2u)$$

$$y^{ve}_{i,\in,h,k,t} \le \frac{1}{2}(x_{i,t,h,k} + x_{t,\in,lch(t),rch(t)}) \qquad (3u)$$

$$x_{i,j,h,k} \le \sum_{l=h}^{k-1} y^{ho}_{i,j,h,l,k} + \sum_{t \in an(j)-\{i\}} y^{ve}_{i,j,h,k,t} \quad \text{for all } i,\ j \in I(T_{i,\in,h,k}),\ h \le k \in ch(i) \qquad (2-3t)$$

$$y^{ho}_{i,j,h,l,k} \le \frac{1}{2}(x_{i,\in,h,l} + x_{i,j,l} + 1, k) \qquad (2t)$$

$$y^{ho}_{i,j,h,l,k} \le \frac{1}{2}(x_{i,j,h,l} + x_{i,\in,l} + 1, k) \qquad (2t')$$

$$y^{ve}_{i,j,h,k,t} \le \frac{1}{2}(x_{i,t,h,k} + x_{t,j,lch(t),rch(t)}) \qquad (3t)$$

$$z_u \ge \frac{1}{n} \sum_{\{i,j,h,k:es(T_{i,j,h,k})=u\}} x_{i,j,h,k} \quad \text{for all distinct Euler strings } u \text{ of } T_{i,j,h,k} \qquad (4)$$

$$\sum_u z_u = m \qquad (5)$$

where $I(T_{i\lfloor,h,k})$ denotes a set of internal vertices that are vertices (neither root or leaves) in $T_{i\lfloor,h,k}$, $an(t)$ denotes a set of ancestors of i ($i \notin an(i)$, and suppose $an(\lfloor) = \varnothing$), and $es(T)$ denotes the Euler string of the ordered tree T (for a tagged tree, the tagged edge with label A is transformed into $AxA\bar{}$, where x is a special symbol representing the tag). It should be noted that if $es(T) = es(T')$, T is isomorphic to T'. In the above program, each variable of $x_{i,j,h,k}$, $y^{ho}_{i,j,h,l,k}$, $y^{ve}_{i,j,h,k,t}$ and z_u takes either 0 or 1. Each $x_{i,j,h,k}$ corresponds to subtree $T_{i,j,h,k}$, and $x_{i,j,h,k} = 1$ iff there exists a nonterminal $A_{i,j,h,k}$ in G that generates $T_{i,j,h,k}$. $z_u = 1$ iff there exists a nonterminal symbol in G that generates subtree u of $T(V, E)$. The meaning of each (in)equality is as follows (Figure 4):

(1u,t) each untagged (tagged) edge $T_{i\lfloor,j,j}$ ($T_{i,j,j,j}$) must be generated,

(2-3u,t) if $A_{i,j,h,k}$ appears in G either for at least some l, $A_{i,j,h,k} \to A_{i,j,h,l}A_{i,j,l+1,k} \in G$, or, for at least some t, $A_{i,j,h,k} \to A_{i,t,h,l}A_{t,j,lch(t),rch(t)} \in G$ holds. $y^{ho}_{i,j,h,l,k} = 1$ means that $T_{i,j,h,k}$ is horizontally divided at root i into the children $\{s \lfloor ch(i) | s \le l\}$ and $\{s \lfloor ch(i) | s > l\}$. $y^{ve}_{i,j,h,k,t} = 1$ means that $T_{i,j,h,k}$ is vertically divided at t into two subtrees $T_{i,t,h,k}$ and $T_{t,j,lch(t),rch(t)}$ (if $j \ne \in$, t must be in $an(j)$, otherwise, a divided tree would have two tags),

(4) $A_{i,j,h,k}$ and $A_{i',j',h',k'}$ are identified if both generate the same Euler string u, and

(5) the number of nonterminal symbols used in G must be m.

IP formulation for unordered trees

In some cases, a given ordered tree is not well compressed. Figure 5 shows an example of such a tree $T(V, E)$, where edges (1, 2),(1, 3),(1, 6),(3, 4), and (3, 5) $\in E$ are labeled with a,c,b,a, and b, respectively. A subtree $T_{3\lfloor,4,5}$ is the same as a subtree with root i. However, we cannot divide the tree into such a subtree and the remaining part, as in the figure, according to production rules in EOTGs. Therefore, we need to extend the above IP for the ordered trees to that for the unordered trees. For this purpose, we extend the EOTG to a grammar for the unordered trees, called the elementary unordered tree grammar (EUTG) [10], and use a simple EUTG for rooted unordered tree compression.

A simple EUTG (SEUTG) is defined as 4-tuple (Σ, Γ, S, Δ) in a similar way to EOTG. A set of production rules Δ is also the same as that of EOTG (Figure 2), except that trees appeared in the production rules are dealt as unordered trees. In other words, there is no sibling relationship between children B and C in the rules (R2u,t). Therefore, we must consider the subtrees of $T(V, E)$ as $T_{i,t,C}$ (Figure 6), where C ($\ne \varnothing$) is a subset of the children of i. Although $ch(i)$ is considered to be the sequence of children of i for the ordered trees, we allow it be a set of the children of i for the unordered trees.

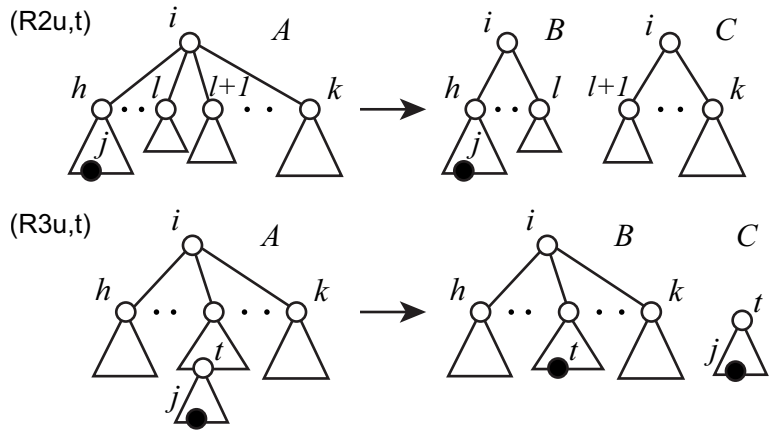

Figure 4 Illustration for bisections of the tagged and untagged ordered trees for the production rules Illustration for bisections of the tagged and untagged ordered trees for the production rules (R2u,t) and (R3u,t) of simple EOTG. *A*, *B*, and *C* correspond to nonterminal symbols in the production rules of Figure 2.

Thus, within the class of SEUTGs, we define the minimum SEUTG problem as follows:

Minimum SEUTG

Input: Rooted unordered tree *T(V, E)* and integer *m*, where *V* is a set of vertices and *E* is a set of labeled edges.

Output: Simple EUTG with *m* nonterminal symbols that generates only *T*.

We must identify unordered subtrees to count the number of nonterminal symbols *m*. For this purpose, we also use the Euler strings *es*(*T*) for the unordered trees *T* as in the minimum SEOTG. First, the unordered tree *T* is transformed into the ordered tree *T'* as follows. The children of each vertex in *T* are sorted by labels, and if it contains a tag, the tag is moved to the first of the children. Next, *es(T)* is calculated to be *es*(*T'*).

Thus, this problem is transformed into the following integer program, where $x_{1,\in,ch(1)} = 1$ holds iff there exists a required EUTG *G:*

Maximize $x_{1\not\in,ch(1)}$

Subject to

$$x_{i,\in,\{j\}} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|\,ch(j)\,| = 0) \tag{1u}$$

$$x_{i,j,\{j\}} = 1 \quad \text{for all } i,\ j \in ch(i)\ (|\,ch(j)\,| > 0) \tag{1t}$$

$$x_{i,\in,C} \le \sum_{C_1(\neq\not\in)\subset C} y^{ho}_{i,\in,C_1,C-C_1} + \sum_{t\in I(T_{i,\in,C})} y^{ve}_{i,\in,C,t} \quad \text{for all } i,\ C \subseteq ch(i) \tag{2-3u}$$

$$y^{ho}_{i,\in,C_1,C_2} \le \frac{1}{2}(x_{i,\in,C_1} + x_{i,\in,C_2}) \tag{2u}$$

$$y^{ve}_{i,\in,C,t} \le \frac{1}{2}(x_{i,t,C} + x_{t,\in,ch(t)}) \tag{3u}$$

$$x_{i,j,C} \le \sum_{C_1(\neq\not\in)\subset C} y^{ho}_{i,j,C_1,C-C_1} + \sum_{t\in an(j)-\{i\}} y^{ve}_{i,j,C,t} \quad \text{for all } i,\ j \in I(T_{i,\in,C}),\ C \subseteq ch(i) \tag{2-3t}$$

$$y^{ho}_{i,j,C_1,C_2} \le \frac{1}{2}(x_{i,j,C_1} + x_{i,\in,C_2})(j \in T_{i,\in,C_1}) \tag{2t}$$

$$y^{ve}_{i,j,C,t} \le \frac{1}{2}(x_{i,t,C} + x_{t,j,ch(t)}) \tag{3t}$$

$$Z_u \ge \frac{1}{n}\sum_{\{i,j,C:es(T_{i,j,C})=u\}} x_{i,j,C} \quad \text{for all distinct Euler strings } u \text{ of } T_{i,j,C} \tag{4}$$

$$\sum_u z_u = m \tag{5}$$

Computational experiments

We implemented the above mentioned IP-based methods for the ordered and unordered trees to perform some computational experiments. We used ILOG

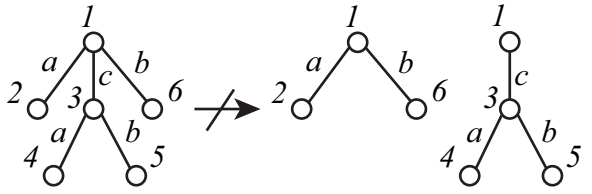

Figure 5 Example of an ordered tree that is not well compressed The left ordered tree cannot be divided into the two right trees. However, this is possible if the left tree is an unordered tree.

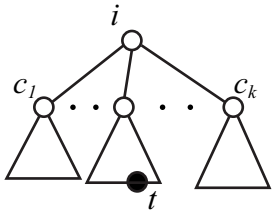

Figure 6 Example of the unordered tree $T_{i,t,C}$ Example of the unordered tree $T_{i,t,C}$ with a set of children of *i*, $C = \{c1, ..., c_k\}$.

CPLEX (version 11.2, http://www.ilog.com/products/cplex/) to solve the integer programs. All of the computational experiments were conducted on a PC with a Xeon CPU 3.33 GHz and 10 GB RAM running under the LINUX OS. In our implementation, we first transformed the minimum grammar problem of the ordered and unordered trees into the integer programs. Next, we used ILOG CPLEX, and obtained the number of nonterminal symbols needed in the minimum grammars of this tree compression. Finally, as the results, the minimum grammars for the tree compression were constructed from the solution of the IP. We also tested the computational time of solving these integer programs. We performed experiments on both artificial data and the glycan tree-structure data, and compared our proposed methods with an existing method.

Artificial data

We chose the left tree T of Figure 5 in which edges labeled with "a" and "b" are connected to both endpoints of an edge labeled with "c", and performed computational experiments, where the simple tree T was treated either as an ordered or an unordered tree. When T was regarded as an ordered tree, we generated the integer program with 13 nonterminal symbols for 9 horizontal and 4 vertical divisions. The number of nonterminal symbols needed in the minimum grammar of T is 7 because the number of production rules except (R1u,t) is 4 and the number of terminal symbols is 3. (Figure 7, in which nonterminal symbols, S,A,..., and F are used). The minimum grammar constructed from the solution of IP is as follows.

$T_{1\swarrow,2,6} \rightarrow T_{1\swarrow,2,2}T_{1\swarrow,3,6}$ (1)
$T_{1\swarrow,3,6} \rightarrow T_{1\swarrow,3,3}T_{1\swarrow,6,6}$ (2)
$T_{1\swarrow,3,3} \rightarrow T_{1,3,3,3}T_{3\swarrow,4,5}$ (3)
$T_{3,\in,4,5} \rightarrow T_{3,\in,4,4}T_{3,\in,5,5}$. (4)

The production rules of this tree compression are also shown in Figure 7. The elapsed time to solve the IP was 0.014 s.

When T was regarded as an unordered tree, we generated the integer program with 14 nonterminal symbols for 12 horizontal and 4 vertical divisions. The minimum number of nonterminal symbols of T is 6 (Figure 8). The minimum grammar was constructed as follows.

$T_{1\swarrow,2,3,6} \rightarrow T_{1\swarrow,3}T_{1\swarrow,2,6}$ (5)
$T_{1\swarrow,3} \rightarrow T_{1,3,3}T_{3\swarrow,4,5}$ (6)
$T_{1\swarrow,2,6} \rightarrow T_{1\swarrow,2}T_{1\swarrow,6}$ (7)
$T_{3\swarrow,4,5} \rightarrow T_{3\swarrow,4}T_{3\swarrow,5}$. (8)

The production rules of this tree compression of T are also shown in Figure 8. The elapsed time to solve the IP was 0.016 s.

In addition to this simple example, we performed experiments for two types of trees with more vertices (Figure 9), where the number of vertices and degree was up to 61 and 20, respectively, and measured the elapsed times. Type A trees only contain vertices with the degree at most two and edges labeled with a, while Type B trees contain edges labeled with a and b, and the height is two. Table 1 shows the results on the elapsed time (seconds) to solve the minimum SEOTG and SEUTG problems by using CPLEX for the ordered and unordered trees of Type A and B with several sizes. m was the same as the minimum number of nonterminal symbols, except the case of Type A trees with 51

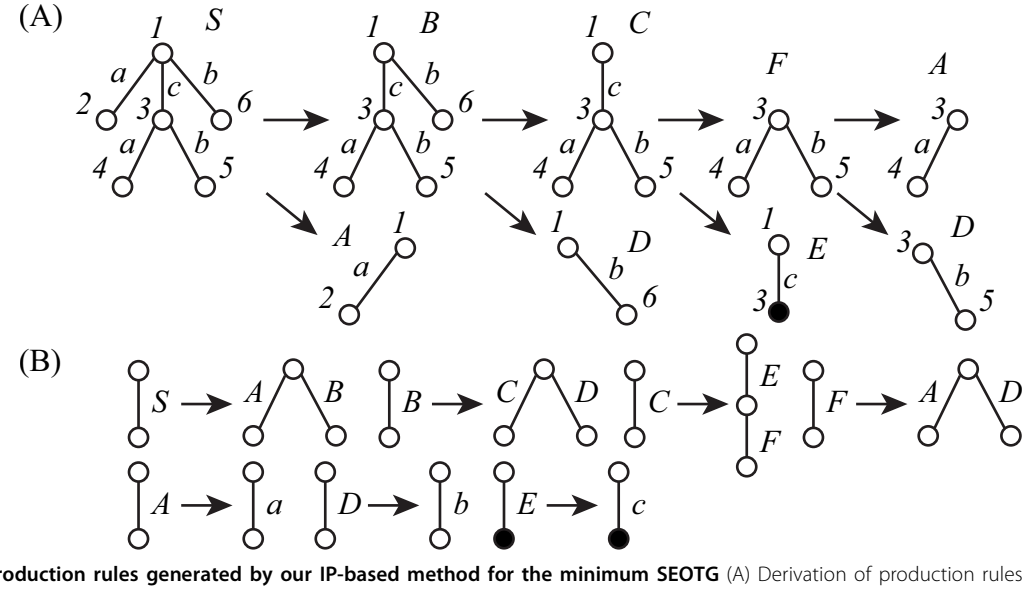

Figure 7 Production rules generated by our IP-based method for the minimum SEOTG (A) Derivation of production rules generated by our IP-based method for the minimum SEOTG. (B) Production rules generated by our IP-based method for the minimum SEOTG.

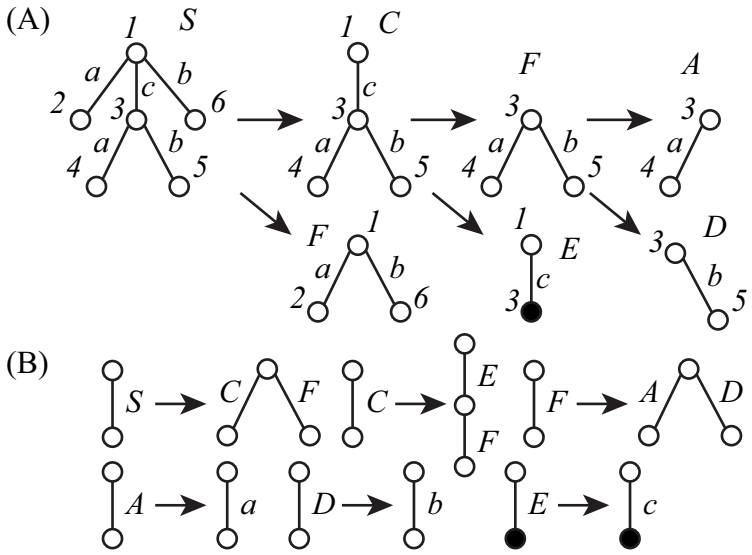

Figure 8 Production rules generated by our IP-based method for the minimum SEUTG (A) Derivation of production rules generated by our IP-based method for the minimum SEUTG. (B) Production rules generated by our IP-based method for the minimum SEUTG.

vertices. In these cases, CPLEX did not output the solution for m = 11 within 8 h. However, we were able to generate the production rules for m = 12, although 10 is the minimum number of nonterminal symbols. If we do not need the minimum grammar, then we can obtain the production rules faster than in the case of finding the minimum grammar. Furthermore, the results show that the elapsed time for an ordered Type A tree was almost the same as that for the corresponding unordered tree, and the time for an ordered Type B tree was shorter than that for the corresponding unordered tree. Even for the ordered tree with 61 vertices, the time was a few minutes. These results suggest that our proposed method is efficient for ordered trees. These results also suggest that the IP-based method for unordered trees should be used when sibling relationships do not have any meanings and the number of vertices and the maximum degree are not so large because the minimum SEUTG size is always smaller than the minimum SEOTG size. However, if the maximum degree is large and sufficient time is not given, the IP-based method for ordered trees should be used. It is because solving the minimum SEUTG problem for such trees may take too much time whereas the method for ordered trees is expected in many cases to provide a small grammar whose size is close to or the same as that of the smallest grammar obtained by the method for unordered trees.

Glycan tree-structure data

It is known that glycans play important roles in a cell such as cellular adhesion and antigen-antibody reaction. Therefore, it is important to analyze structures of glycans. Hizukuri *et al.* extracted characteristic functional motifs of glycans, predicted a leukemia specific glycan motif, and confirmed by biological experiments that the *Agrocybe cylindracea* galectin specifically recognized human leukemic cells [11]. Thus, it is also important to find motifs and repeated patterns of glycans. We obtained twelve glycans, G02703, G03655, G03710, G04045, G04458, G04666, G04859, G05058, G05256,

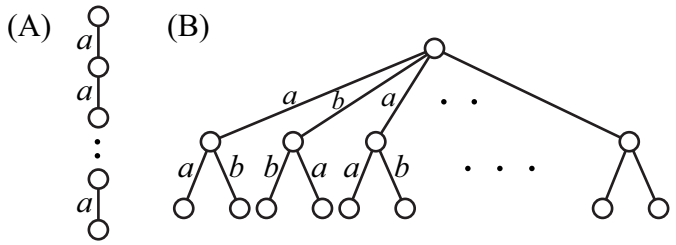

Figure 9 Trees used in experiments for evaluation of our IP-based methods (A) Trees having only vertices with degree at the most two. (B) Trees having vertices with degree more than two.

Table 1 Results on the elapsed time (seconds) for ordered and unordered trees of type A and B

tree type	max degree	# vertices	ordered		unordered	
			m	time	*m*	time
A	2	11	7	0.021	7	0.019
A	2	31	10	302.74	10	329.20
A	2	41	10	8063.19	10	7730.64
A	2	51	12*	230.51	12*	233.44
B	3	7	9	0.011	8	0.010
B	6	19	11	0.185	10	1.108
B	8	25	11	1.404	10	26440.01
B	10	31	12	2.265	-	-
B	16	49	11	481.15	-	-
B	20	61	13	432.72	-	-

Results on the elapsed time (seconds) for ordered and unordered trees of type A and B (in Figure 9) with several sizes. m was the same as the minimum number of nonterminal symbols for each tree, except the case denoted by '*'. '-' denotes that the solver took more than 8 hours.

G05552, G06867, and G09054 as rooted trees from the KEGG Glycan database [12]. We labeled each edge with a lower-case letter corresponding to the type of sugar of the lower endpoint, because the edges are not labeled in the original data. For each glycan, the maximum degree, the number of vertices, and the number of distinct labels are shown in Table 2. Then, we applied our proposed IP-based method for SEUTG to each glycan as an unordered tree, and obtained the production rules. Figures 10, 11, 12, and 13 show extracted patterns from the production rules of G03655, G04458, G04666, and G05058. We can see from the result of the generated production rule that the tree of G03655 contains 2 of the same subtrees with 4 vertices and 3 of the same subtrees with 5 vertices, the tree of G04458 contains 2 of the same subtrees with 8 vertices and the subtree contains 3 of the same subtrees with 3 vertices, the tree of G04666 contains 3 of the same subtrees with 5 vertices and 2 of the same subtree with 3 vertices, and the tree of G05058 contains 3 of the same subtrees with 6 vertices and 2 of the same subtrees with 5 vertices. We were able to extract patterns similar to those of G03655, G04458, G04666, G05058 for the other glycans. The detailed derivation diagrams of production rules for the four glycans are available on our supplementary web site (http://sunflower.kuicr.kyoto-u.ac.jp/morihiro/treegram/).

We compared the results of the grammar size for the minimum SEOTG and SEUTG by our methods with those of an existing method, TREE-BISECTION [10]. TREE-BISECTION repeatedly divides a given tree horizontally and vertically such that the size of a divided subtree is similar to that of another subtree until each subtree consists of an edge. It is known that TREE-BISECTION computes in polynomial time a simple EOTG of size $O(mn^{5/6})$ [10], where m is the size of the minimum simple EOTG and n is the number of vertices of the given tree. Table 2 shows the results of the grammar size and the elapsed time by our proposed IP-based methods for the minimum SEOTG and SEUTG problems, and TREE-BISECTION. The minimum SEOTG size was the same as that of the minimum SEUTG for each glycan except G05552 because the tree contains vertices only with at most two children, and all subtrees of a vertex having three children are isomorphic. The size of the grammar generated by

Table 2 Statistics of glycans, G02703, G03655, G03710, G04045, G04458, G04666, G04859, G05058,G05256, G05552, G06867, and G09054, and results on the grammar size

glycan	max degree	# vertices	# distinct labels	Min SEOTG		Min SEUTG		TREE-BISECTION	
				size	time	size	time	size	time
G02703	3	26	3	22	3.68	22	2.9	32	0.001
G03655	3	34	3	47	0.96	47	2.32	49	0.001
G03710	3	28	3	20	0.47	20	0.51	20	0.001
G04045	3	36	3	20	1.77	20	1.98	22	0.001
G04458	3	21	2	16	1.55	16	0.69	36	0.001
G04666	3	20	4	25	1.41	25	0.94	33	0.001
G04859	3	19	5	27	0.12	27	0.25	29	0.001
G05058	3	25	5	26	3.03	26	66.28	36	0.001
G05256	3	25	2	19	3.14	19	3.98	29	0.001
G05552	3	19	5	27	0.66	23	0.23	27	0.001
G06867	3	28	3	22	2.22	22	6.46	26	0.001
G09054	4	31	5	29	2.81	29	6.71	29	0.001

Statistics of glycans, G02703, G03655, G03710, G04045, G04458, G04666, G04859, G05058, G05256, G05552, G06867, and G09054, and results on the grammar size and the elapsed time (seconds) by our proposed IP-based methods for the minimum SEOTG and SEUTG problems, and TREE-BISECTION [10].

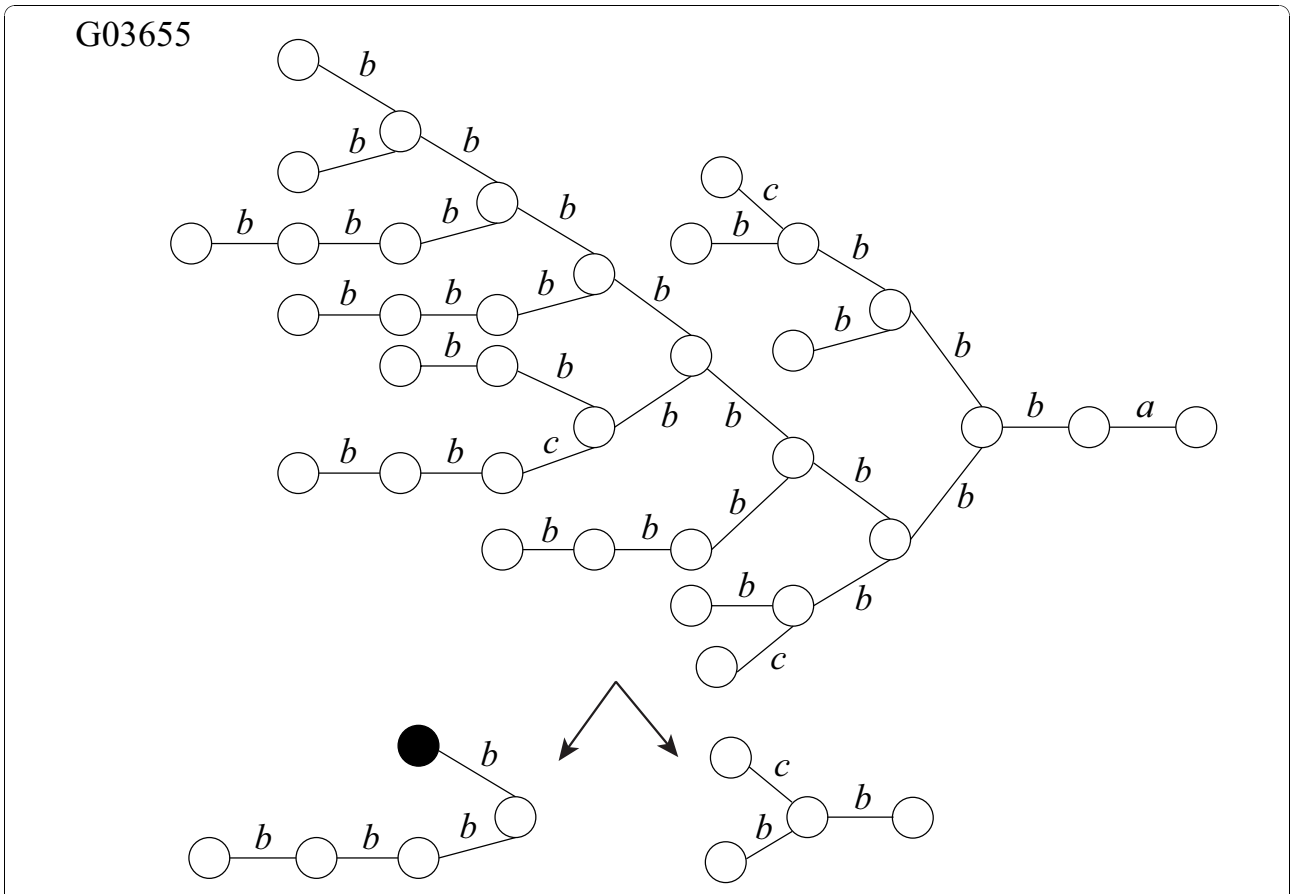

Figure 10 Extracted patterns from glycan G03655 The label related with the lower endpoint is attached to each edge. Labels, *a*, *b*, and *c* denote GlcNAc, Man, and P, respectively.

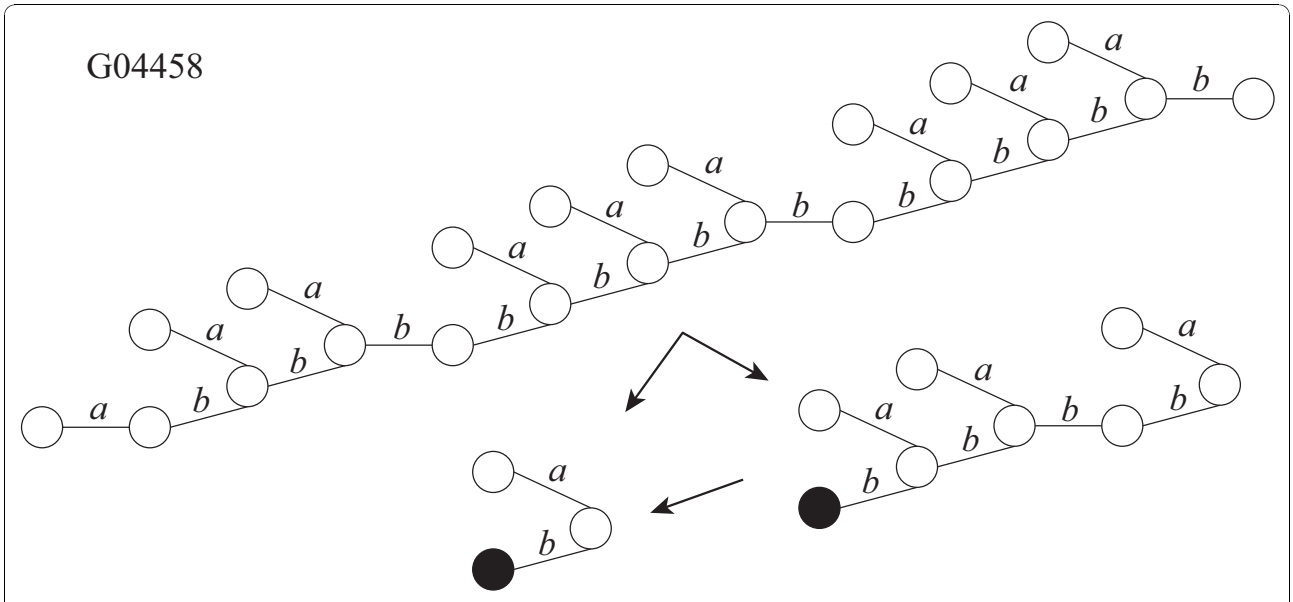

Figure 11 Extracted patterns from glycan G04458 The label related with the lower endpoint is attached to each edge. Labels, *a*, and *b* denote Xyl, and Glc, respectively.

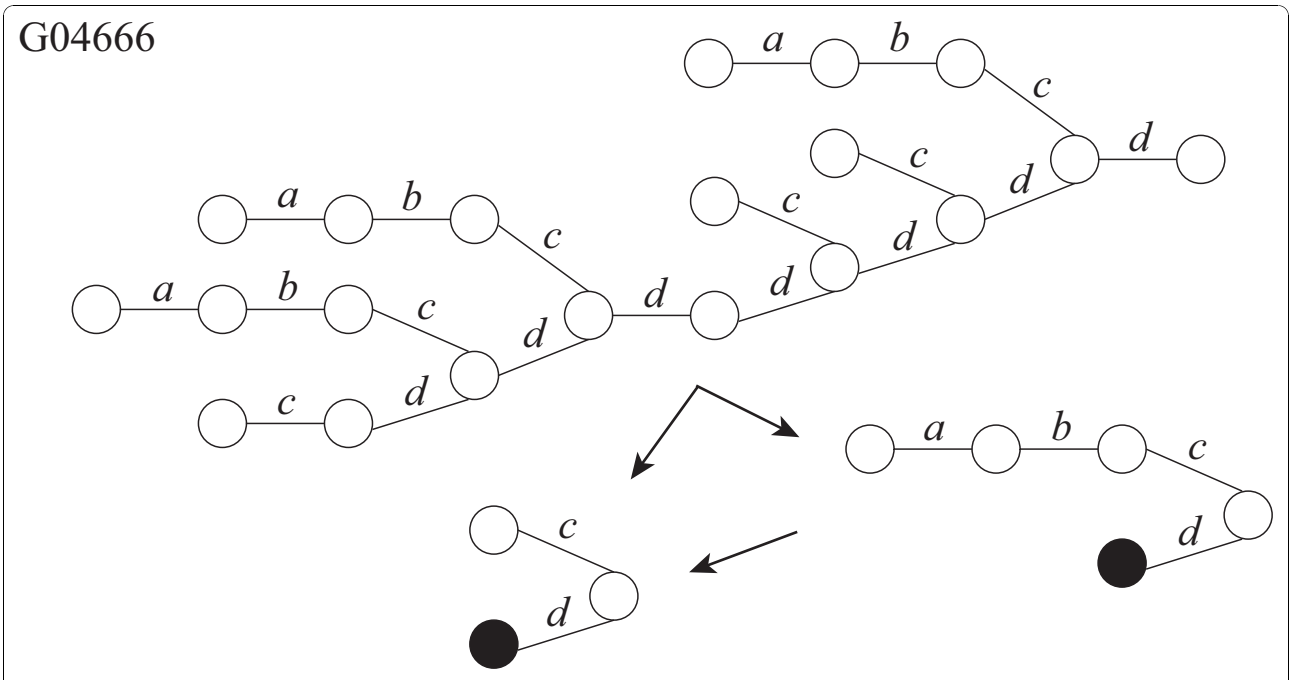

Figure 12 Extracted patterns from glycan G04666 The label related with the lower endpoint is attached to each edge. Labels, *a*, *b*, *c*, and *d* denote LFuc, Gal, Xyl, and Glc, respectively.

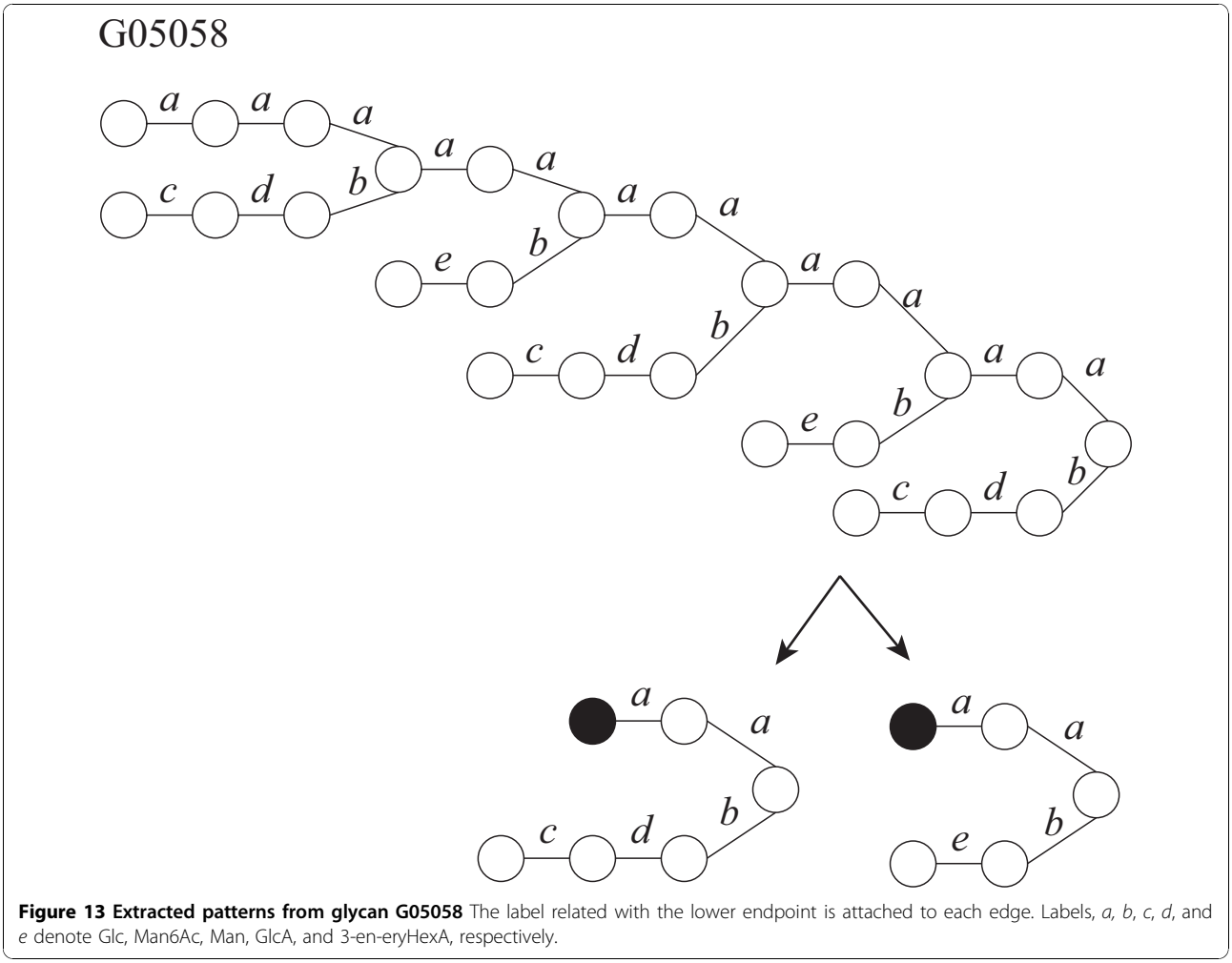

Figure 13 Extracted patterns from glycan G05058 The label related with the lower endpoint is attached to each edge. Labels, *a*, *b*, *c*, *d*, and *e* denote Glc, Man6Ac, Man, GlcA, and 3-en-eryHexA, respectively.

our methods was always smaller than or equal to that by TREE-BISECTION, and the ratio was 1.0 (G09054) to 2.25 (G04458). This result shows that our proposed method performs better with the compression ratio than TREE-BISECTION.

Conclusions

We proposed integer programming-based methods for finding the minimum grammars to generate given strings, ordered trees, and unordered trees. By conducting computational experiments, we confirmed that our IP formulations work correctly. The results also show that our IP-based grammar compression is efficient for ordered trees, although some improvements are required for unordered trees.

We applied our proposed method to glycan tree-structure data, and extracted interesting patterns. Although these patterns were obtained from production rules generated for a single tree, we may be able to extract common patterns and rules from multiple glycans by extending our methods to find minimum grammars to generate given forests.

In this paper, we dealt with grammars for trees. However, real structured data often contain some cycles. Therefore, we are in the process of developing IP-based methods for more complex structured data.

Acknowledgements

This work was partially supported by Grants-in-Aid #22240009 and #21700323 from MEXT, Japan.
This article has been published as part of *BMC Bioinformatics* Volume 11 Supplement 11, 2010: Proceedings of the 21st International Conference on Genome Informatics (GIW2010). The full contents of the supplement are available online at http://www.biomedcentral.com/1471-2105/11?issue=S11.

Authors contributions

TA gave the basic idea. YZ and MH developed and implemented the algorithms, and carried out the experiments. YZ, MH, and TA authored and approved the manuscript.

Competing interests

The authors declare that they have no competing interests.

Published: 14 December 2010

References

1. Li M, Badger J, Chen X, Kwong S, Kearney P, Zhang H: **An information-based sequence distance and its application to whole mitochondrial genome phylogeny.** *Bioinformatics* 2001, **17**:149-154.
2. Hayashida M, Akutsu T: **Image compression-based approach to measuring the similarity of protein structures.** *In Proc. 6th Asia-Pacific Bioinformatics Conference* 2008, 221-230.
3. Hayashida M, Akutsu T: **Comparing biological networks via graph compression.** *BMC Systems Biology* 2010, **4(Supp 2)**:S13.
4. Charikar M, Lehman E, Liu D, Panigrahy R, Prabhakaran M, Sahai A, Shelat A: **The smallest grammar problem.** *IEEE Transactions on Information Theory* 2005, **51**:2554-2576.
5. Rytter W: **Application of Lempel-Ziv factorization to the approximation of grammar-based compression.** *Theoretical Computer Science* 2003, **302**:211-222.
6. Sakamoto H, Maruyama S, Kida T, Shimozono S: **A space-saving approximation algorithm for grammar-based compression.** *IEICE Transactions on Information and Systems* 2009, **92-D**:158-165.
7. Busatto G, Lohrey M, Maneth S: **Efficient memory representation of XML document trees.** *Information Systems* 2008, **33**:456-474.
8. Murakami S, Doi K, Yamamoto A: **Finding frequent patterns from compressed tree-structure data.** *In Proc. 11th Int. Conf. Discovery Science* 2008, 284-295.
9. Yamagata K, Uchida T, Shoudai T, Nakamura Y: **An effective grammar-based compression algorithm for tree structured data.** *In Proc. 13th Int. Inductive Logic Programming* 2003, 383-400.
10. Akutsu T: **A bisection algorithm for grammar-based compression of ordered trees.** *Information Processing Letters* 2010, **110**:815-820.
11. Hizukuri Y, Yamanishi Y, Nakamura O, Yagi F, Goto S, Kanehisa M: **Extraction of leukemia specific glycan motifs in humans by computational glycomics.** *Carbohydrate Research* 2005, **340**:2270-2278.
12. Hashimoto K, Goto S, Kawano S, Aoki-Kinoshita K, Ueda N, Hamajima M, Kawasaki T, Kanehisa M: **KEGG as a glycome informatics resource.** *Glycobiology* 2006, **16(5)**:63R-70R.

doi:10.1186/1471-2105-11-S11-S4
Cite this article as: Zhao *et al.*: Integer programming-based method for grammar-based tree compression and its application to pattern extraction of glycan tree structures. *BMC Bioinformatics* 2010 **11**(Suppl 11):S4.